Popular Computing

Volume 8

Number 12

93

December 1980

The 7s Problem

1	7	38	770070
2	70	39	70707
3	777	40	7000
4	700	41	77777
5	70	42	7770
6	7770	43	7707707
7	7	44	7700
8	7000	45	7777777770
9	777777777	46	7707070
10	70	47	70077
11	77	48	7770000
12	77700	49	7007
13	7007	50	700
14	70	51	700077
15	7770	52	700700
16	70000	53	700077
17	77707	54	77077777770
18	7777777770	55	770
19	77007	56	7000
20	700	57	77007
21	777	58	77077070
22	770	59	77077777
23	770707	60	77700
24	777000	61	700707
25	700	62	7770770
26	70070	63	777777777
27	7707777777	64	7000000
28	700	65	70070
29	7707707	66	7777770
30	7770	67	7707077
31	777077	68	7770700
32	700000	69	70000707
33	777777	70	70
34	777070	71	70077
35	70	72	777777777000
36	77777777700	73	70007
37	777	74	7770

An Old Problem

Any integer, K, divides some number, N, that consists entirely of the digits 0 and 7. That fact was proved by David Ferguson in our issue number 37, April 1976.*

Clearly, for each K there are infinitely many N's. The table on the cover shows the smallest value of N for the first 74 values of K. The table was calculated by Mark Lewis Spitz, Reseda, California.

Mr. Spitz built up dividends as binary numbers, then changed each 1 to a 7, and tested the resulting number for divisibility by K. By incorporating several shortcuts (e.g., factors of 2 or 5 in K force a low-order zero in N), the table was produced quickly.

The solution is a brute force one, however, and would become impractical for many larger values of K (one notes that multiples of 9 appear to be particularly difficult).

So our Problem 280 is to devise a more efficient algorithm to extend the table. □

*The cover of that issue contained the prediction that pocket computers would be available for $250 by September of 1980. Fred Gluckson points out that we were off by one month and one dollar: the Radio Shack pocket machine sells for $249.

Publisher: Audrey Gruenberger
Editor: Fred Gruenberger
Associate Editors: David Babcock, Irwin Greenwald, Patrick Hall
Contributing Editors: Richard Andree, William C. McGee, Thomas R. Parkin, Edward Ryan
Art Director: John G. Scott
Business Manager: Ben Moore

POPULAR COMPUTING is published monthly at Box 272, Calabasas, California 91302. Subscription rate in the United States is $20.50 per year, or $17.50 if remittance accompanies the order. For Canada and Mexico, add $1.50 per year. For all other countries, add $3.50 per year. Back issues $2.50 each. Copyright 1980 by POPULAR COMPUTING.

In our issue number 69 we discussed this problem:

> For any positive integral value of R, what is the smallest power of 2 that contains R embedded contiguous zeros?

The status of the problem (which dates back to 1951) at that time was this:

R	X	
1	10	
2	53	
3	242	
4	373	
5	1491	
6	1492	
7	6801	erroneously listed as 6081
8	14007	
9	unknown, but greater than 60000.	

The entry for case 9 can now be filled in: 100823. In this number of 30351 digits, the nine contiguous zeros occupy the 21896 through 21904 positions. The computer run to gain these facts executed over 53 billion instructions.

The single- and double-digit distributions of the digits of the 100823rd power of 2 are given.

2936	2895	3136	3140	3063	2953	3131	3060	3041	2996
0	1	2	3	4	5	6	7	8	9

00	285	17	270	34	316	51	285	68	310	85	308
01	281	18	292	35	319	52	312	69	318	86	312
02	294	19	292	36	313	53	305	70	293	87	313
03	299	20	267	37	312	54	315	71	296	88	303
04	317	21	277	38	334	55	260	72	305	89	289
05	266	22	311	39	309	56	316	73	331	90	314
06	301	23	341	40	321	57	308	74	344	91	274
07	281	24	310	41	300	58	276	75	294	92	308
08	301	25	347	42	313	59	295	76	303	93	298
09	311	26	331	43	303	60	313	77	323	94	293
10	279	27	327	44	282	61	307	78	289	95	284
11	288	28	301	45	299	62	337	79	282	96	336
12	297	29	324	46	324	63	335	80	281	97	309
13	304	30	302	47	291	64	306	81	289	98	313
14	281	31	298	48	321	65	302	82	350	99	267
15	274	32	309	49	309	66	277	83	296		
16	318	33	328	50	281	67	326	84	299		

+4	+5	+1	+2	+3	+3	-10	-8	+3	+2	+2	+2	+2	-3	+3	+10	-10	-10	+1	+2
+2	-3	-4	-3	-2	+8	+6	+2	+10	+2	+2	-5	-3	-3	-5	+7	-7	+7	+2	+2
+2	-1	-5	+2	+5	+5	+5	+3	-8	-5	+10	+10	+10	+10	+5	-10	-5	-2	-3	-10
+2	-1	+5	+2	+3	+5	+3	+3	-3	-3	-3	-3	+1	+10	+5	+5	-5	-5	-3	+2
+1	+4	+4	+2	+4	-2	-5	-5	-5	+1	+1	+2	+1	+5	-3	+3	+3	+3	-3	+2
-5	-5	-5	-5	-3	+2	-10	+2	+2	+1	+2	+2	+2	-10	-2	+2	-5	-5	-3	+1
+10	-2	-2	+6	-10	-10	+3	-7	-7	-7	-9	+11	+2	+1	+10	-7	-7	-7	+11	+1
+2	-2	+6	+6	+9	+7	-3	-3	-3	-2	+2	+2	+1	-5	-5	+1	+1	+2	-8	+2
-1	-1	+1	+4	+6	-4	-6	-8	-10	-2	-6	+2	+2	-10	-4	-10	+13	+2	-11	-11
+5	+2	+4	-10	-7	-4	-1	+6	+6	+6	-3	-3	-3	+5	+5	+5	+7	+1	-9	-9
+1	+1	+8	+1	+1	+1	-10	+8	+3	+3	+3	+3	+3	+3	-2	-2	-6	+1	+1	+3
+3	+5	+3	-5	-10	-5	-5	-5	+6	-20	+5	-11	-5	-5	-5	+3	+3	+3	+2	-10
+3	+5	-11	-5	+2	+10	+10	-1	-1	+4	-2	+4	+11	+2	+2	+4	+1	+9	+9	-5
-1	-1	+2	-4	-4	-4	+10	+10	+4	+4	-2	-5	-5	+9	+8	+10	+6	+4	+4	-1
+4	+3	+1	+1	-4	+3	+3	+10	+10	+10	+10	+2	-5	-5	-3	-2	+6	-3	-3	-1

Traverse

The 10 x 20 grid on the facing page contains a collection of more-or-less random plus and minus numbers.

A path is to be found, starting at the upper left cell and arriving eventually at the lower left cell. This path is to proceed only orthogonally (that is, never diagonally); it must never cross itself; and is to have the property that the progressive totals of the numbers in the cells that it passes through are always composite numbers (that is, the sum must never be prime).

For example, if the path begins by going straight down, the progressive totals would be 4, 6, 8, 10, 11, and the 11 is not allowed, being prime.

Similarly, if the path begins by going straight across to the right, the progressive totals of the numbers that are crossed would be 4, 9, 10, 12, 13, and 13 is prime.

A total of zero should never be reached, and the number <u>one</u> is not prime for this exercise.

> Note: the given pattern was constructed by first making a path that met the given conditions, and then filling in all the other cells with attractive looking numbers. Thus, there could be correct paths other than the one we had in mind.

So we know a priori that the problem has at least one solution. That leaves only two other problems:

1. What is a possible algorithm for finding <u>any</u> solution?

2. Is it a suitable computer problem?

PROBLEM 281

BOOK REVIEW

The 6502 Microprocessor

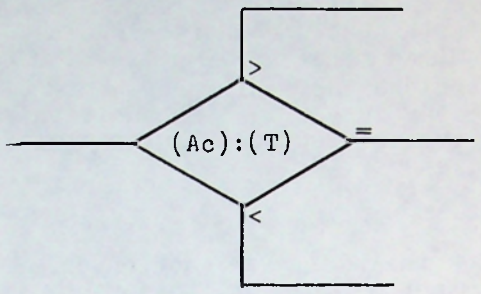
If you have an Apple, or a PET, or an AIM or KIM, then you have a 6502 microprocessor and you can write and execute programs in 6502 language. Usually, this is done via an assembler, since controlling the machine in absolute hex is possible, but it can get tedious and full of errors. You also need--most of the time--a good reference book, to explain the fine points of the processor, if only to refresh your memory. We will discuss here three such books:

First, there is the MOS Microcomputing Programming Manual (for which no author is credited), put out in 1976 by MOS Technology, Inc., the makers of the processor.

Then there is Programming the 6502 by Rodnay Zaks, published by Sybex in 1978.

Finally, there is 6502 Assembly Language Programming by Lance Leventhal, published by Osborne/McGraw-Hill in 1979.

The edition of Zaks' book that is reviewed here came with an errata list of a hundred or so errors; this printed list should have one more entry: it is labelled "ERRATUM." In addition to the errors acknowledged in the list, the book is riddled with typos. Zaks' book is completely useless (worse, highly misleading) unless those errata are inserted into the text, but even then it is not too helpful. For example, the 6502 has the operation CMP, which is the COMPARE instruction. To be precise, it is a COMPARE ACCUMULATOR TO STORAGE type instruction, as in:



where (Ac) is the contents of the accumulator and (T) is the contents of the operand (say, a word of storage). The use of CMP normally calls for taking one of the following paths, for which the simplest operations are given here:

$A = T$	BEQ
$A \neq T$	BNE
$A < T$	BMI
$A \geq T$	BPL
$A > T$	BNE followed by BPL
$A \leq T$	BEQ followed by BMI

It would be difficult to deduce all of this from the description given in Zaks' book.

Each of the three books lists all the op-codes in the 6502's repertoire; this is most likely the part of the book that the user will consult most frequently. It is illuminating to examine how each of the books treats an op-code. Picking one at random, let's see how ASL (Shift Left One Bit) is handled by each book.

The MOS book shows pictorially that a zero is generated into bit zero of the addressed word, and that bit 7 is moved to C (Carry). The N, Z, and C bits are enabled.

The ASL operation has 5 different op-codes, to permit zero page addressing, indexed addressing, or accumulator addressing. These are given, together with the assembly language form, and the number of execution cycles.

Zaks' book gives all the same information, but in a form that is much more difficult to read. For example, the op-code for ASL with absolute addressing is 0E. Zaks shows that all the ASL commands have the form 000bbb10, and that the form for absolute addressing has bbb = 011, from which one may deduce the code 0E. In addition, Zaks gives the data path for each op-code, but the diagrams require considerable study.

Leventhal's book includes, for each op-code, all the information one might need about it (some of it a bit difficult to dig out, especially when the list is not consulted frequently), but with one thing the other two books lack; namely, examples of the use of the command under various conditions.

The MOS book gives the bare facts with each op-code, and references back to the text material for more involved information.

Let's track another item through the three books. The 6502 has two index registers, called X and Y. They have similar uses and are controlled similarly, but

THEY ARE NOT INTERCHANGABLE.

The two most important differences are these:

1. Certain commands (such as the shift commands and the increment and decrement commands) are indexable only with X, not with Y.

2. The firmware of some microcomputers (specifically the Apple) uses the Y register and does not store and restore it. Thus, if the user has a loop controlled with the Y register and in that loop invokes any of the firmware, the program will do funny things.

In an 8-bit machine, good coding will make extensive use of the index registers. The novice, or the casual user, will expect much tutorial material on index registers, as well as all the facts about their use in practice. The MOS book does indeed have a long discussion about indexing, but it is only tantalizing about the X and Y registers of the 6502:

> "Both index registers have the ability to be compared to memory (CPX, CPY) and to be incremented (INX, INY) and decremented (DEX, DEY). Because of OP CODE limitations, X and Y have slightly different uses. X is a little more flexible because it has Zero Page operations which Y does not have with exception of LDX and STX. Aside from which modes they modify, the registers are autonomous, independent and of equal value."

Zaks' book devotes a short paragraph to index registers; it says mainly that they are good things to have, and that they will be considered only in conjunction with indirect addressing.

Leventhal's book does not list index registers in its index and there is no specific section in the book on the subject of indexing. Leventhal does, however, tell more about the distinction between X and Y than either of the other two books:

> "Only the X index register can be used for pre-indexed indirect addressing...Only the Y index register can be used with post-indexed indirect addressing...X and Y are not interchangeable since no instructions have both forms of simple indexing with both X and Y. In fact, the only instructions which allow zero-page indexing with Y are LDX and STX."

Zaks' book can be dismissed as fairly useless. For serious work with the 6502, the choice (if one must be made) is between the MOS book and Leventhal's. The former deals mainly with the architecture and pathology of the micro-processor; the latter, as its name implies, deals much more with assembly language programming. Leventhal's book refers incessantly to a prior book by Osborne (An Introduction to Microcomputing).

All three books are infuriating for avoiding the things the user wants most often. It is all very well and good to be given a tight loop to clear 100 words in page zero; what the user probably needs is a good routine to clear 10,000 words in general. Even elementary address modification:

```
INC   LOW       |    LDA   LOW
BNE             |    BNE
INC   HIGH      |    DEC   HIGH
                |    DEC   LOW
```

is buried somewhere in all three books. Leventhal's book is rich with practical examples, but every one of them is written in page zero, which gets annoying. No one writes programs completely in page zero. To be sure, the user can figure out for himself how to rewrite the routines in upper memory, but the authors should do some of this.

Still, if a choice must be made, the recommendation is for Leventhal's book. The price is unknown, due to a phobia among publishers to reveal the retail price of a book to potential customers.

Problem numbers in recent issues were omitted as follows:

277	The Y-Sequence Issue 90, page 20.
278	Givoli's Problem Issue 91, page 14.
279	Newdice Toss Issue 91, page 16.

Speaking of Records...

In issue 91 we reported on the calculation of factorial 11000, which was claimed as a record and was imprudently touted as "...in one sense, the...largest number ever calculated."

This brought a response from Harry Nelson, editor of the Journal of Recreational Mathematics and co-holder of the record for the world's largest known prime.

First, Mr. Nelson called attention to a report in 1965 in Mathematical Tables and other Aids to Computation, in a review by Dan Shanks, citing a solution to Archimedes' cattle problem consisting of a 206,545-digit integer. The solution is in the Unpublished Mathematical Tables file of MTAC.

Second, Mr. Nelson revealed that he had calculated every thousandth factorial from (1000!) to (1,000,000!) and sent along the first few digits of each of them, together with their size. Factorial ONE MILLION has 5,565,709 digits, and begins with 826393168...

His result for (11000!) agrees nicely with ours. Our calculation for the one minuscule factorial took 42 hours, while his calculation took around 10.5 hours for all thousand results.

At the time we first commented on Nelson's notable calculation of the 27th Mersenne prime (in our issue 81), we reprinted a letter that had appeared in the Los Angeles Times from a professor of astronomy, deploring the waste of (government) money and effort. Here is the reply that was sent to the professor from the public relations office at the Livermore laboratory:

> I must agree with you that prime numbers don't have many practical applications. There are some, as Mr. Harry Nelson, one of the scientists here who discovered the 27th Mersenne Prime, pointed out recently. He told me that only three years ago there seemed to be no use for prime numbers as large as 100 digits. However, since then, scientists have found them useful in coding messages and the idea may one day be used in the country's national security program. Mostly, though, the hunt for larger prime numbers seems to be an exercise in basic scientific research.

You seemed concerned that time and money might have been wasted while our researchers looked for their prime number. I share your interest in frugality at government institutions. However, I do want to reassure you that in this case no time and money was wasted; the researcher's discovery was simply a delightful spinoff of work they would have been doing anyway.

The news story you read was probably abbreviated from the attached press release. The release explains that Mr. Nelson's job here is to test new computers for accuracy and reliability. When the Lab received a new Cray-1 computer last February, Mr. Nelson used David Slowinski's computer program to search for prime numbers as part of the test for the computer. In the process the new prime number was found. Mr. Nelson tells me that if he hadn't used Mr. Slowinski's program he would have used a similar diagnostic program with no practical value except to test the computer. As it turned out, the two researchers were able to make an interesting discovery while doing routine work, at no extra cost to the taxpayer. I wish that all scientific research could be so economical.

Sincerely,
Linda Currey
Public Information Office

Nelson himself offers these comments:

As to the present record lasting several decades, that depends upon the sequence of primes. If there is another Mersenne prime less than 100000, it will be found within two years, in my opinion. The program Slowinski and I wrote can be greatly improved (now that we are working at truly long precision) by fast-multiply methods. We know of one such method, easily applied to numbers in the 50000-bit range, which will result in an improvement of at least 3 and perhaps 5 over the version used to find M_{44497}.

Incidentally, we stopped at 50023. Reiterating my main point above, it won't surprise me in the least if the primeness or compositeness of all Mersenne numbers up to 100000 is determined by the end of 1981. Media interest (including that of POPULAR COMPUTING) has an effect on such things.

and finally, to make the adventure complete, we have this reaction from the astronomy professor:

> I read the news story in the Los Angeles Times when we were working on the budget for the School of Natural Sciences and literally trying to make ends meet. And if I recall correctly, the Times article led one to believe that two computer experts worked for three months to discover a 13,395-digit prime number. That prompted my letter to the Times.
>
> With my past experiences with press releases, I should have been more careful; they are notorious for being inaccurate.
>
> I was also aware of the possibility to use prime numbers in coding messages, but that morning it simply did not come to mind.
>
> I hope my letter to the Times did not cause any inconvenience, and I certainly apologize for my hasty remarks. I am certain Mr. Nelson and Mr. Slowinski are performing good, basic research work; and I wish we had at California State University, Long Beach, a public information office staffed with someone with Ms Linda Currey's ability.

Answer to Last Month's Challenge Problem:

Most people are astonished to find that there is a tremendous difference which method of repayment is used and that the difference is relatively greater as the amount of the loan increases. One has only to calculate the interest on two loans--of $1000, and of $1 000 000, say--for an odd number of years to see clearly what takes place.

Square Root Again

Webb Simmons, San Diego, California, suggests the use of the following algorithm for square root:

```
                 A = Argument         0 ≤ A ≤ 1
                 S = Square root
                 D = Delta

            Set S = 1/4

            Set D = 1/4

LØØP:       IF S > A  GØTØ NEXT

            A - S ⟶ A

            S + D + D ⟶ S
              *


NEXT        D/2 ⟶ D        (Right shift, not divide)

            S - D ⟶ S

            A + A ⟶ A

            IF  D > 0 GØTØ LØØP  **
```

The algorithm will usually terminate at the point marked (*), with the required root in S, controlled by a loop counter. The test at (**) is an alternative. If the procedure is controlled by a loop counter, then the test at (**) would be replaced by an unconditional branch to LØØP. The algorithm uses only add, subtract, shift, compare, and branch--all fast operations on any machine.

Mr. Simmons, who declares himself self-taught in computing, comments on his algorithm as follows:

On a computer, large or small, whose instruction set is limited to addition and subtraction for its mathematics, one must perform division by the well-known shift, test, and (possibly) subtract algorithm that develops the quotient one bit at a time. The square root algorithm above has its only utility in such a context; it develops the square root one bit at a time, and this only slightly slower than performing just one divide. Most assuredly my algorithm is not intended for use with a high level language except for demonstration purposes, and it would not be used in any language on computers like the Univac 1110, IBM 370, or CDC Cyber, etc., that has fast floating point hardware.

However, in a floating point computer environment, a square root routine often has a component module that takes the square root of a scaled value less than one. The argument to this module is derived from the mantissa part of the full argument that is supplied to the square root subroutine.

My first exposure to a computer square root method was on the Univac 1108 which had fast floating hardware. Its floating number is composed of a biased power of 2 as its characteristic (also called the power, or exponent) and a scaled fraction as its mantissa (also called factor or coefficient). The mantissa is always less than one but not less than one-half. For an odd characteristic, the square root algorithm increases the characteristic by one and divides the mantissa by 2; thus, the effective argument to the central square root algorithm is the adjusted mantissa: $1/4 \leq$ value < 1. The 1108 evaluates a polynomial to obtain a first guess with about 9 bits of accuracy; it then uses Heron's rule two times without further testing to obtain a full precision, normalized 27 bit mantissa (its word size is 36 bits with a 9 bit characteristic). The adjusted (possibly incremented) characteristic is then divided by 2 (except for the bias) and combined with the mantissa for the returned value.

My next exposure to the square root function on computers was on a Nova minicomputer which had no multiply or divide hardware. Its programmers decided to take a square root in much the same manner as on the 1108, but it was distressingly slow. I then devised an earlier version of the algorithm; I have placed this on a PDP-11 (without floating hardware) and on a Z80 micro.

Along the way, I have had occasion to use Heron's rule, which can be derived as follows.

Assume that $x^2 \cong N$. If ϵ is the error, then

$$N \equiv (x + \epsilon)^2 = x^2 + 2\epsilon x + \epsilon^2 = x^2 + 2\epsilon x.$$

Solving for ϵ: $\epsilon = (N - x^2)/2x$

we should get a better approximation to $\sqrt{N} \cong x + \epsilon$

$$x_{n+1} = x_n + \frac{N - x_n^2}{2x_n} = x_n + (1/2)\left[\frac{N}{x_n} - x_n\right]$$

It is very interesting that you refer to my algorithm as "cleverly disguised" interval-halving. The action of my algorithm is best seen in binary, where the root is developed one bit at a time in much the same manner as a division. It is not interval-halving, but I suspect that the error decreases at about the same rate on the average because, at any time, the error is about 1/2 bit in the next bit position being developed. Lacking floating hardware, Newton's method cannot possibly be faster because you will need, usually, to or more divisions, and my method is almost as fast as one division.

Lately I have become interested in fast functional algorithms. I firmly believe that there are better ways than the classical methods introduced mainly by Hastings and his followers. For example, the CORDIC algorithm for trig functions (used on H-P and TI pocket calculators) is a real winner.

CORDIC means COrdinate Rotation DIgital Computer, but there is no need for the coordinates to rotate. The rotation had appeal in the context in which it was first employed (as a substitute for analog "resolvers" in aircraft navigational computers) but for ordinary math I have worked out a simpler approach in which the world sits still while a vector rotates. It is my guess that CORDIC has been neglected by the big computer world because (1) they had something already that worked, and (2) CORDIC is always presented, for some reason, in a strange, needless complicated manner.

Note: In our issue number 20, we presented a baker's dozen distinct algorithms for square root. Mr. Simmons seems to have given us a 14th method. We invite comments on it. □

Problem Solution

Givoli's problem (number 278, in issue 91) called for finding the smallest progressive total of the primes (this is a sequence that begins 2, 5, 10, 17, 28, 41,...) that is a multiple of each successive integer.

For example, for the integer 11, the 8th term of the sequence is a multiple (77), but for the integer 12 it is necessary to go to the 97th term to find the first exact multiple (22548). It is still an open question as to whether or not such a multiple can always be found.

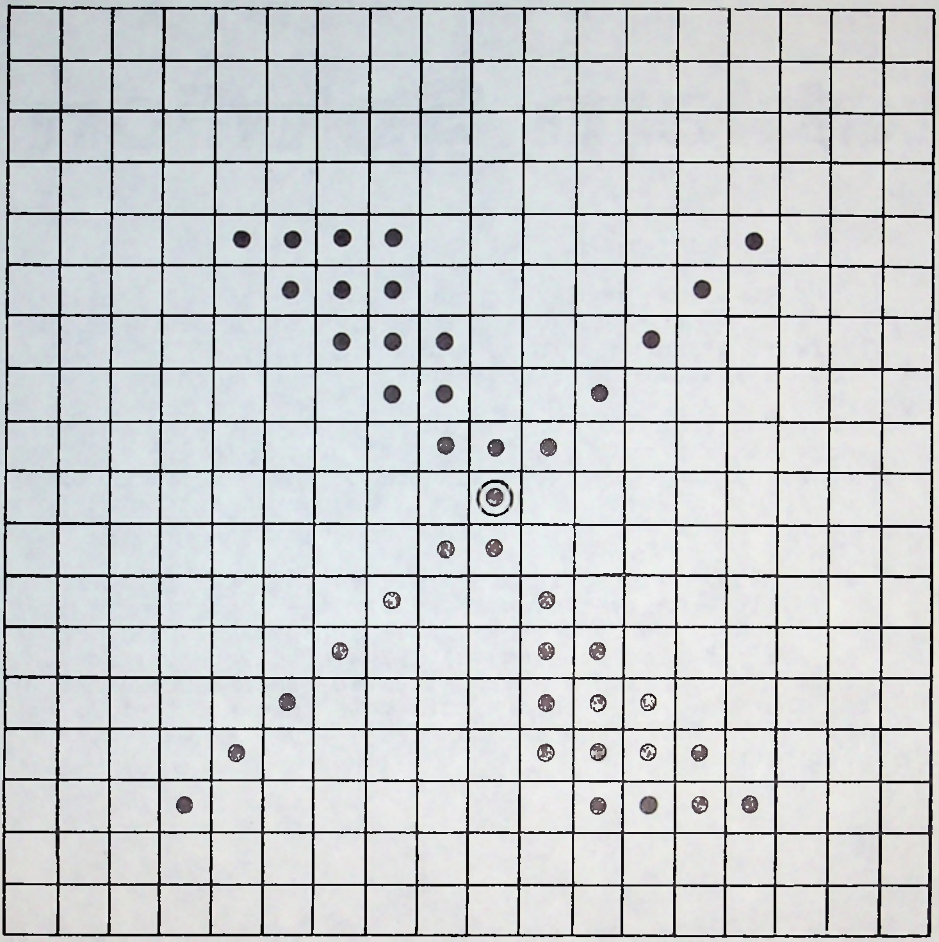
It was stated in the article about the problem that no multiple had yet been found for the integers 303 or 326. The author of the problem, Nahman Givoli, of Tel Aviv, had noted that multiples of 6 were particularly difficult to calculate, and the two numbers mentioned were the last for which the results were unknown (that is, as far as the work had been carried), and they were not multiples of 6.

Ted Raab, Clifton Park, New York, reports that those entries to our table Y are now known:

d	N	S
303	1229	5,736,396
326	1765	12,461,676

Raab's calculations were done in Applesoft on a 16K Apple II, in less than 40 minutes.

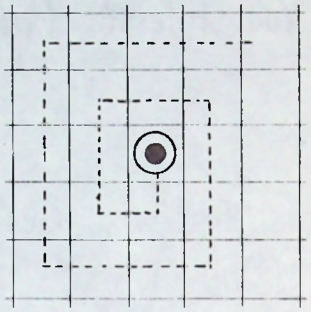
□



A Pattern Of New Distances

In the grid of squares shown on the facing page, a pattern is formed by the following scheme:

The center square (shown circled) is selected. The other squares in the grid are then sampled in spiral fashion:



For each square sampled, the distance to all previously selected squares is calculated. If <u>any</u> of these distances is new, then the square is selected, marked as shown, and entered into a table of selected squares. For those squares for which <u>all</u> the distances to <u>all</u> previously selected squares are already in the list of distances, the square is by-passed.

An endless series of such patterns can be created (and creating them is an excellent coding exercise in almost any computer language), by varying:

1. The scanning scheme. For the pattern shown, a clockwise spiral scan was used. One can imagine diagonal scans, TV-type scans, and so on.

2. The pattern can be "seeded" with additional fixed selected squares.

For computers whose normal output is to a screen, the production of such patterns can be plotted as the points are selected. For each pattern, the first few points are found rapidly, but the time for each successive point gets longer.

PROBLEM **282**

Inventory Clearance Sale

Limited Time Only

Complete Your File of Popular Computing

On orders received up to January 31, 1981:

1. For 10 or more of the issues still available.
2. 50¢ (U.S.) per copy.
3. Chosen from issues 1 through 82 only.
4. Mailed to one address by surface mail.

JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG	SEP	OCT	NOV	DEC		
			1	2	3	4	5	~~6~~	~~7~~	~~8~~	9	Vol. 1	1973
10	11	12	~~13~~	14	~~15~~	~~16~~	~~17~~	~~18~~	19	~~20~~	21	Vol. 2	1974
22	23	24	25	26	27	28	~~29~~	30	31	32	33	Vol. 3	1975
~~34~~	~~35~~	36	37	38	39	40	41	42	43	44	45	Vol. 4	1976
46	47	48	49	50	51	52	53	54	55	56	57	Vol. 5	1977
58	59	60	61	62	63	64	65	66	67	68	69	Vol. 6	1978
70	71	72	73	74	75	76	77	78	79	80	81	Vol. 7	1979
82	83	84	85	86	87	88	89	90	91	92	93	Vol. 8	1980